A Recommendation System Framework to Generalize AutoRec and Neural Collaborative Filtering

Ramin Raziperchikolaei
Rakuten, Inc.
San Mateo, CA, USA
ramin.raziperchikola@rakuten.com

Young-joo Chung
Rakuten, Inc.
San Mateo, CA, USA
youngjoo.chung@rakuten.com

***Abstract*—AutoRec and neural collaborative filtering (NCF) are two widely used neural network-based frameworks in the recommendation system literature. In this paper, we show that these two apparently very different frameworks have a lot in common. We propose a general neural network-based framework, which gives us flexibility in choosing elements in the input sources, prediction functions, etc. Then, we show that AutoRec and NCF are special forms of our generalized framework. In our experimental results, first, we compare different variants of NCF and Autorec. Then, we indicate that it is necessary to use our general framework since there is no specific structure that performs well in all datasets. Finally, we show that by choosing the right elements, our framework outperforms the state-of-the-art methods with complicated structures.**

***Index Terms*—Collaborative filtering, autoencoders**

I. Introduction

Neural networks have been playing an important part in the recommendation system papers recently [1, 2, 3, 4, 5, 6, 7]. They have been used to extract user and item representations and to model the complicated relationship between them. The two widely used structures are autoencoders [8] and neural collaborative filtering (NCF) [4, 5, 6, 7].

The first autoencoder-based recommendation system, known as AutoRec, was proposed in [8]. In AutoRec, the autoencoder tries to reconstruct the user (or item) interaction vector in training. At the test time, the reconstructed values are considered as the prediction of the unknown values of the interaction vector.

Neural collaborative filtering (NCF) was proposed in [4] to improve the performance of the matrix factorization (MF). The idea is to use neural networks, instead of the dot product, to convert the user and item representations to their interaction value.

Several variants of Autorec [9, 10, 11, 12] and NCF [5, 6, 7] have been proposed to improve the performance using different architectures, inputs, loss functions, and regularizers. However, there is no systematic comparison between them. In other words, it's not clear what makes one structure perform better than the other one in different datasets.

In this paper, we propose a general framework that unifies variants of NCF and AutoRec. This framework enables us to conduct systematic comparison of the state-of-the-art structures, to identify the effect of the changes in the framework, and to shed a light on directions for novel variants of this family of methods. Our contributions are as follows:

- We propose a general framework for neural network-based collaborative filtering (Section III). As shown in Figure 1, our framework consists of three modules: 1) the representation learning module, 2) the fusion module, and 3) the prediction module. By modularizing the RS model, our framework allows the flexible choice (tuning) for each module to find the best model for given applications.
- We show that AutoRec and NCF, two major neural network-based CF architectures that seems very different at first glance, are the special cases of this framework (Section IV and V). With this, we demonstrate in Table I that their variants can be categorized under our framework by choosing different elements in each module.
- We investigate the impact of different modules in our structure, such as inputs and loss functions, on recommendation tasks using three datasets (Section VI). The results reveal that no specific structure performs best in all the datasets, which shows the necessity of using our general framework to select the best module based on the final tasks. The results also show that our framework outperforms SOTA thanks to the flexibility in choosing the elements.

II. Related works

In matrix factorization (MF), the dot product is used to model the interaction between the users and items from their embeddings. Neural collaborative filtering (NCF) [4] is proposed to improve the performance of the MF by replacing the dot product by more complicated functions. In [4], GMF and MLP were proposed as two models from the NCF framework. In GMF [4], a nonlinear layer takes the element-wise product of the user and item embeddings and predicts their interaction value. In MLP [4], an MLP takes the concatenation of the user and item embeddings and predicts their interaction value. In [5], CFNet-rl and CFNet-ml were proposed, which are very similar to the GMF and MLP in [4]. The only difference is

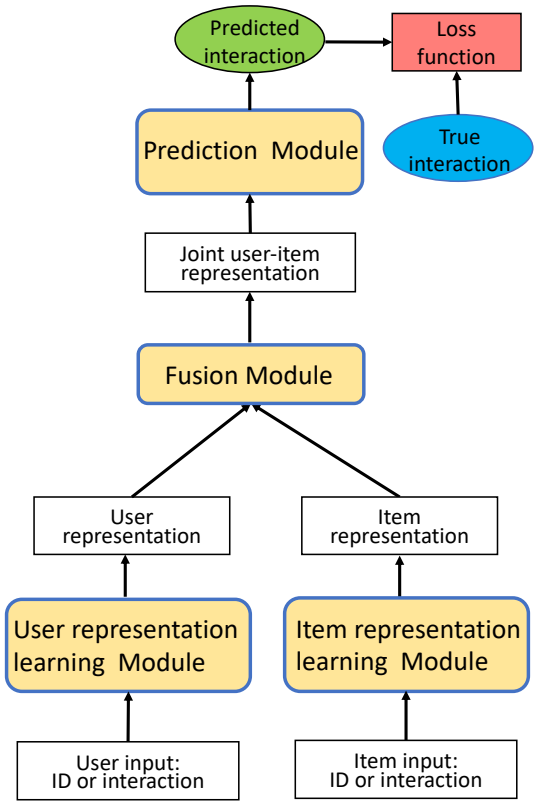

Fig. 1. Our general framework. MF, NCF, and AutoRec are a special form of this structure.

that CFNet uses interaction vector and MLPs to extract user and item representations, while [4] uses embedding layers. In [6], MLPs are used to extract user and item representations and their cosine similarity is used to predict the interactions.

AutoRec[8] is the first autoencoder-based recommendation method. The idea is to reconstruct the interaction vectors and use the reconstructed values as the predicted interactions. In [9], the side information is added to the AutoRec structure. In [10], the AutoRec structure is used for implicit feedback prediction. In [11], the user reviews and attention mechanism are used to improve the prediction performance. In [12], a denoising autoencoder is proposed which is capable of eliminating the overfitting towards identity. There are many other works that use autoencoders in recommendation systems [9, 13, 14, 15, 16, 17].

III. Our General Framework

Let us introduce the notations first. We denote the sparse interaction matrix by $\mathbf{R} \in \mathbb{R}^{m \times n}$, where m and n are the number of users and items, respectively, $R_{jk} > 0$ is the interaction value of the user j on the item k, and $R_{jk} = 0$ means the interaction is unknown. The goal is to predict the unknown interactions in $\mathbf{R}$. The ith row of a matrix $\mathbf{H}$ is shown by $\mathbf{H}_{i,:}$ and the jth column is shown by $\mathbf{H}_{:,j}$.

In Fig. 1, we show the structure of our framework, which contains two inputs, three modules, and a loss function. The *user and item representation learning modules* take the user and item inputs and generate user and item representations, respectively. The *fusion module* combines the user and item representations and generates the joint representation. Finally, the *prediction module* takes the joint representation and predicts the interaction value. In the rest of this section, we will explain the task of each module of this framework in more details.

A. Input layer

ID [4, 7, 18] and interaction vector [5, 6, 15, 17] are the two widely used inputs in the recommendation system literature:

- In case of ID as the input, each user j is represented by an m-dimensional vector $\mathbf{I}^u_j$, which is 0 everywhere except in the jth index, which is 1. We define the kth item's input vector in the same way and denote it by $\mathbf{I}^i_k$, which is n-dimensional.
- In case of the interaction vector as the input, the jth row and the kth column of the interaction matrix are the jth user and kth item input vectors, which are denoted by $\mathbf{R}_{j,:}$ and $(\mathbf{R}_{:,k})^T$, respectively.

B. Representation learning module

The two widely used learning modules are embedding layers [4, 7, 18] and multi-layer perceptrons (MLPs) [5, 6], which are explained in the following:

- Embedding layers are used to generate representations from user and item IDs. Let us denote the user embedding matrix as $\mathbf{E}^u \in \mathbb{R}^{m \times d}$. Then the d-dimensional representation of the jth user is defined as $\mathbf{z}^u_j = \mathbf{I}^u_j \mathbf{E}^u$, which gives us the jth rows of the embedding matrix. In the same way, we define the item embedding matrix as $\mathbf{E}^i \in \mathbb{R}^{n \times d}$ and the the d-dimensional item representation of the kth item as $\mathbf{z}^i_k$.
- MLPs are used to generate representations from user and item interactions. For the jth user, a user MLP $\mathbf{g}^u()$ takes the interaction vector of the jth user and generates the d^u-dimensional user representation $\mathbf{z}^u_j = \mathbf{g}^u(\mathbf{R}_{j,:})$. In the same way, the d^i-dimensional representation of the kth item is defined as $\mathbf{z}^i_k = \mathbf{g}^i((\mathbf{R}_{:,k})^T)$.

C. Fusion module

This module takes the d^u dimensional representation of the jth user ($\mathbf{z}^u_j$) and d^i dimensional representation of the kth item ($\mathbf{z}^i_k$), and generates a d dimensional joint user and item representation, denoted by $\mathbf{z}_{jk}$. *Concatenation* and *element-wise product* are two popular ways to combine the representations [4, 5]. In the case of element-wise product, the assumption is that the user and item representations have the same dimension, i.e. $d^u = d^i$.

D. Prediction module

Prediction module takes the joint representation $\mathbf{z}_{jk}$ and outputs the predicted interaction value $\hat{R}_{jk}$ of the user j on item k, i.e., $\hat{R}_{jk} = h(\mathbf{z}_{jk})$. MLPs can be used as the function $h()$ to model the complicated relations between users and items.

TABLE I

HOW TO SET THE INPUT LAYERS, MODULES, AND LOSS FUNCTIONS IN OUR FRAMEWORK TO ACHIEVE DIFFERENT NCF-BASED AND AUTOREC-BASED MODELS. U-INPUT AND U-REP REFER TO USER INPUT AND USER REPRESENTATION, I-INP AND I-REP REFER TO ITEM INPUT AND ITEM REPRESENTATION, E-W REFERS TO ELEMENT-WISE PRODUCT, AND NON-LIN REFERS TO ONE NONLINEAR LAYER.

methods/modules	u-input	u-rep learning	i-input	i-rep learning	fusion	prediction	loss
MF	ID	Embedding	ID	Embedding	e-w product	linear	mse
GMF [4]	ID	Embedding	ID	Embedding	e-w product	non-lin	mse
MLP [4]	ID	Embedding	ID	Embedding	concatenation	MLPs	mse
CFNet-rl [5]	interaction	MLPs	interaction	MLPs	e-w product	non-lin	BCE
CFNet-ml [5]	interaction	linear	interaction	linear	concatenation	MLPs	BCE
DMF [6]	interaction	MLPs	interaction	MLPs	normalized e-w product	linear	normalized BCE
ONCF [7]	ID	Embedding	ID	Embedding	outer-product	CNNs	BPR[19]
U-AutoRec [8]	ID	Embedding	interaction	MLPs	e-w product	non-lin	mse
I-AutoRec [8]	interaction	MLPs	ID	Embedding	e-w product	non-lin	mse

a) Dot product in our framework.: Dot product has been used frequently in the literature to model the relationship between the users and items [6, 13, 15, 18, 19]. In our framework, we can achieve the dot product by using the element-wise product as the fusion module and a linear layer as the prediction module.

E. Loss function

Various loss functions, including Mean squared error (MSE) and Binary cross entropy loss (BCE), have been used in the literature. Let us assume the training set T contains a set of pairs of users and items and their interaction values, then two popular loss functions are defined as:

$$l_{\text{mse}} = \sum_{j,k \in T} (\hat{R}_{jk} - R_{jk})^2$$
$$l_{\text{BCE}} = \sum_{j,k \in T} R_{jk} \log(\hat{R}_{jk}) + (1 - R_{jk}) \log(1 - \hat{R}_{jk}). \quad (1)$$

The l_{mse} can be used with both explicit and implicit interactions [4, 6, 15], while the l_{BCE} is mainly used with the implicit interactions [5, 20].

Let us discuss how the training set is created when the interactions are explicit and implicit. In the case of explicit interactions, the training set T contains all the pairs of the users and items with the known interactions. In the implicit interaction prediction, all the known interactions are 1, so training on them will lead to converging to the trivial solution of always returning 1. To prevent converging to this solution, in addition to all the pairs with the known interactions, the set T also contains a subset of the pairs with the unknown interaction with value 0.

IV. NCF UNDER OUR FRAMEWORK

Neural Collaborative Filtering framework utilizes neural networks, instead of the dot product in MF, to model the relationship between the users and items. By selecting different inputs, modules, and loss functions, we can achieve the different NCF models proposed in the previous works, such as GMF [4], MLP [4] , CFNet-rl [5], CFNet-ml [5], and ONCF [7]. Table I contains some of these models.

V. AUTOREC UNDER OUR FRAMEWORK

One of the main contributions of this paper is to show that AutoRec's framework is very similar to the NCF framework. The objective function of the user-based AutoRec (U-AutoRec) is:

$$\sum_{j=1}^{n} ||\mathbf{R}_{j,:} - \hat{\mathbf{R}}_{j,:}||^2_{\mathcal{O}}, \quad \text{s.t.} \quad \hat{\mathbf{R}}_{j,:} = \mathbf{q}^{\text{d}}(\mathbf{q}^{\text{e}}(\mathbf{R}_{j,:})) \quad (2)$$

where $\mathbf{q}^{\text{e}}()$ is the encoder that maps the interaction vector to the low-dimensional representation, and $\mathbf{q}^{\text{d}}()$ is the decoder that reconstructs the interaction vector. $||\cdot||^2_{\mathcal{O}}$ computes the norm over the known ratings and ignores the unknown ones.

The encoder and decoder both have a set of fully connected layers. We expand the decoder, assuming it has L layers, as follows:

$$\hat{\mathbf{R}}_{j,:} = \mathbf{q}^{\text{d}}(\mathbf{q}^{\text{e}}(\mathbf{R}_{j,:})) = \sigma(\dots\sigma(\sigma(\mathbf{q}^{\text{e}}(\mathbf{R}_{j,:})\mathbf{W}^1)\mathbf{W}^2)\dots\mathbf{W}^L) \quad (3)$$

where $\mathbf{W}^1, \dots, \mathbf{W}^L$ are the weights (parameters) of the decoder. For the better analysis, let us denote the output of the $(L-1)$th layer of the decoder by $\mathbf{z}^u_j \in \mathbb{R}^d$. We can rewrite the Eq. (3) as:

$$\hat{\mathbf{R}}_{j,:} = \mathbf{q}^{\text{d}}(\mathbf{q}^{\text{e}}(\mathbf{R}_{j,:})) = \sigma(\mathbf{z}^u_j \mathbf{W}^L), \quad \text{s.t.} \quad \mathbf{z}^u_j = \sigma(\dots\sigma(\sigma(\mathbf{q}^{\text{e}}(\mathbf{R}_{j,:})\mathbf{W}^1)\mathbf{W}^2)\dots\mathbf{W}^{L-1}). \quad (4)$$

The size of the matrix $\mathbf{W}^L$ is d by m, i.e., it has a d-dimensional vector for each of the m items. *So the dimension of the $\mathbf{W}^L$ is identical to the item embedding matrix.*

We rewrite the objective function of Eq. (2) in a format similar to the ones in Eq. (1):

$$\sum_{j=1}^{n} ||\mathbf{R}_{j,:} - \hat{\mathbf{R}}_{j,:}||_{\mathcal{O}}^2 = \sum_{j,k \in T} (\hat{R}_{jk} - R_{jk})^2 \quad \text{s.t.}$$
$$\hat{R}_{jk} = \sigma(\mathbf{z}_j^u (\mathbf{z}_k^i)^T) \quad \mathbf{z}_k^i = \mathbf{W}_{:,k}^L,$$
$$\mathbf{z}_j^u = \sigma(\dots \sigma(\sigma(\mathbf{q}^e(\mathbf{R}_{j,:})\mathbf{W}^1)\mathbf{W}^2) \dots \mathbf{W}^{L-1}) \quad (5)$$

Recall that the training set T contains the pairs with the known ratings. The predicted rating is the dot product (followed by an activation) of the user and item representations. The user representation is the output of the $(L-1)$th layer of the decoder, which is a set of fully connected layers on top of the user interaction vector. The item representation $\mathbf{z}_k^i$ is the kth column of the item embedding matrix $\mathbf{W}^L$. Table I (last two rows) shows how we should set the modules in our framework to make it the same as AutoRec.

VI. Experiments

a) Datasets.: For the explicit feedback prediction, we use ml100k [21], which contains 100 000 ratings from around 1 000 users on 1 600 movies. For the implicit feedback prediction, we use three datasets: 1) Last.fm[1], which contains 69 149 interactions from 1 741 users on 2 665 items, 2) Amazon music (AMusic)[22], which contains 1 700 users, 13 000 items, and 46 000 interactions, and 3) Amazon Toys (AToys)[22] , which contains 3 137 users, 33 953 items, and 84 642 interactions. We use the pre-processed datasets provided in [5], which is publicly available [2].

b) Evaluation metrics.: We report the root mean square error (RMSE) to evaluate the rating prediction performance in the ml100k dataset. We follow [4, 5] to report Hit Ratio (HR) and NDCG to evaluate the implicit feedback prediction performance in the rest of the datasets.

c) Implementation details and source code.: We implemented our method using Keras with TensorFlow 2 backend. We ran all the experiments on a 32GB GPU. For each method, we tried a range of learning rates 10^{-1} to 10^{-4} with SGD and Adam optimizers and picked the one that works best. We used early stopping to avoid overfitting. The source code is available at https://raminrazi.github.io/.

A. NCF vs Autorec: why we need a general framework

In Fig. 2, we show how choosing different elements in our framework impacts the performance. In this figure, results in the left panel use dot product to model the interaction between users and items (element-wise product as the fusion and linear layer as the prediction modules). Results in the right panel use concatenation as the fusion module and MLP as the prediction module. The notation u-id/i-id refers to using user/item ID as the input and user/item embedding as the rep. learning module. The notation u-inter/i-inter refers to using user/item interaction vector as the input and user/item MLPs as the rep. learning

[1]https://www.last.fm

[2]https://github.com/familyld/DeepCF

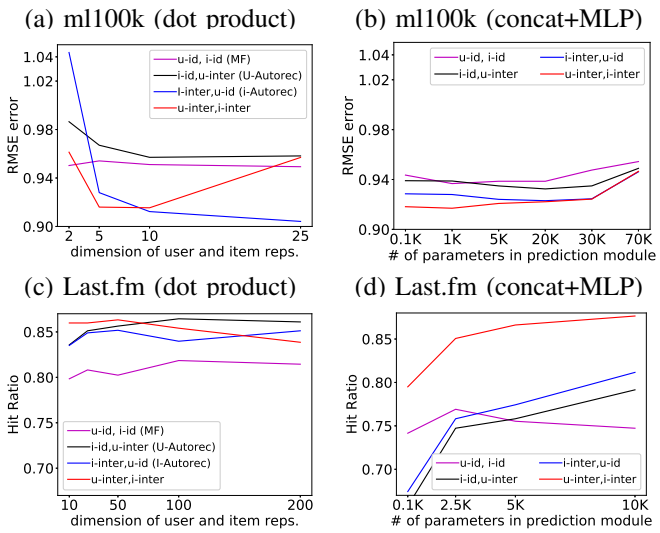

Fig. 2. We compare four different combinations of the user and item inputs and representation learning module. We report RMSE error in ml100k dataset and HR@10 in Last.fm dataset. MF, NCF, and Autorec are instances of our framework.

module. In all plots, the model becomes more complex (more parameters) as we move from left to right of the x-axis.

We compare NCF and Autorec structures using the results of the Fig. 2: the left panel contains the results of the I-Autorec and U-Autorec, and all the curves in the right panel (concat+MLP) are instances of NCF framework. There are more curves corresponding to the NCF than the ones corresponding to the Autorec because the Autorec structure uses specific type of the inputs while NCF can use different variants of inputs. We can also see that I-AutoRec achieves the best results in rating prediction in ml100k while NCF performs best in implicit feedback prediction in Last.fm dataset. So we cannot conclude that one structure is better than the other one in all the datasets.

To understand why neither NCF nor Autorec is always the best choice, we should look at them as the instances of our general framework. It is known that for any machine learning model to perform well, no matter what the application is, one needs to choose the right parameters, such as the number of layers and neurons and activation functions in a neural network. Recommendation system is no exception: representation learning, fusion, and prediction modules are all parameters in a recommendation system model and they should be set carefully from one dataset to the other.

The main advantage of our framework is in separating all the modules in a recommendation system model and letting the user set them to the right elements based on the application. By using our framework, one can perform systematic experiments such as module (parameter)-tuning, and see which module is contributing to or harming the performance in the target dataset. After identifying the module leading to a performance drop, we can replace its element with other ones and get the improvement.

TABLE II
COMPARING OUR FRAMEWORK WITH THE CFNET [5]. OUR FRAMEWORK ACHIEVES BETTER RESULTS THAN CFNET IN ALL THREE DATASETS.

	AMusic		last.fm		AToy	
	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
CFNet	0.382	0.243	0.880	**0.602**	0.362	0.216
Ours	**0.409**	**0.248**	**0.889**	**0.602**	**0.384**	**0.223**

Another advantage of our framework is that by choosing the right elements it can outperform methods with complicated structures in the literature. In Table II, we compared the performance of our general framework with CFNets. We run CFNet (code is publicly available by the authors[3]) and our method three times and put the average in this table.

CFNet [5] is a neural network-based method that combines representations from two simpler models to learn two joint user and item representations. It predict the final interaction by a nonlinear mapping over the concatenated representations.

Our general framework has a simpler structure than CFNet since it learns one representation per user and item and one joint representation. But, at the same time, it's flexible and powerful since we can use different elements in each module.

In all three datasets, our framework uses interaction vectors as the inputs, MLPs as the representation learning, and mse as the loss function. In AMusic and AToy, our framework uses dot product as the fusion function, while in last.fm it uses MLPs. We chose the settings on a very small subset of the dataset and a validation set.

The results show that our framework achieves better results than CFNet in all three datasets of the Table II.

VII. CONCLUSION AND FUTURE WORKS

Neural collaborating filtering and AutoRec are two apparently different neural network-based frameworks in recommendation systems. NCF focuses on learning better representations for the users and items and combining them with fully connected layers. AutoRec reconstructs the interaction vectors to estimate the unknown ones. In this paper, we proposed a general framework with representation learning, fusion, and prediction modules. We showed that AutoRec and different variants of the neural collaborative filtering are special cases of our framework, which can be achieved by selecting specific functions for our modules. Our experiments show that the impact of the functions changes as we change the dataset and the final task, i.e. one set of functions that work well in one dataset, might not be the best option in another dataset.

As future work, we want to expand our framework to cover the new techniques in the recommendation systems, such as the graph neural networks. Also, we want to find the impacts of other fusion functions and loss functions in the prediction performance.

[3]https://github.com/familyld/DeepCF

REFERENCES

[1] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah, "Wide & deep learning for recommender systems," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLSRS)*, 2016.

[2] X. He and T.-S. Chua, "Neural factorization machines for sparse predictive analytics," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2017.

[3] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "DeepFM: a factorization-machine based neural network for CTR prediction," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

[4] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web (WWW)*, 2017.

[5] Z.-H. Dong, L. Huang, C.-D. Wang, J.-H. Lai, and P. S. Yu, "DeepCF: a unified framework of representation learning and matching function learning in recommender system," in *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, 2019.

[6] H.-J. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

[7] X. He, X. Du, X. Wang, F. Tian, J. Tang, and T.-S. Chua, "Outer product-based neural collaborative filtering," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

[8] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "AutoRec: Autoencoders meet collaborative filtering," in *Proceedings of the 24th International Conference on World Wide Web (WWW)*, 2015.

[9] F. Strub, R. Gaudel, and J. Mary, "Hybrid recommender system based on autoencoders," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLSRS)*, 2016.

[10] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-n recommender systems," in *Proceedings of the 9th ACM International Conference on Web Search and Data Mining (WSDM)*, 2016.

[11] J. P. Zhou, Z. Cheng, F. Perez, and M. Volkovs, "TAFA: Two-headed attention fused autoencoder for context-aware recommendations," in *Proceedings of the 14th ACM Conference on Recommender Systems (RecSys)*, 2020.

[12] H. Steck, "Autoencoders that don't overfit towards the identity," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[13] S. Li, J. Kawale, and Y. Fu, "Deep collaborative filtering

via marginalized denoising auto-encoder," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM)*, 2015.

[14] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2015.

[15] X. Dong, L. Yu, Z. Wu, Y. Sun, L. Yuan, and F. Zhang, "A hybrid collaborative filtering model with deep structure for recommender systems," in *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017.

[16] S. Zhang, L. Yao, and X. Xu, "AutoSVD++: An efficient hybrid collaborative filtering model via contractive auto-encoders," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2017.

[17] T. Li, Y. Ma, J. Xu, B. Stenger, C. Liu, and Y. Hirate, "Deep heterogeneous autoencoders for collaborative filtering," in *Proceedings of the 18th IEEE International Conference Data Mining (ICDM)*, 2018.

[18] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[19] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: bayesian personalized ranking from implicit feedback," in *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2009.

[20] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun, "xDeepFM: combining explicit and implicit feature interactions for recommender systems," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2018.

[21] F. M. Harper and J. A. Konstan, "The MovieLens datasets," *ACM Transactions on Interactive Intelligent Systems*, vol. 5, no. 4, pp. 1–19, 2015.

[22] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *Proceedings of the 25th International Conference on World Wide Web (WWW)*, 2016.